U.S. PATENT APPLICATION
FOR A SYSTEM AND METHOD
FOR INSERTION OF MARKERS
INTO A DATA STREAM

ASSIGNEE: iReady Corporation

Attorney Docket No. SPL3004 – P1

Stafford Partners, LLC
40 Lake Bellevue, Suite 100
Bellevue, WA 98005

SYSTEM AND METHOD FOR INSERTION OF MARKERS INTO A DATA STREAM

FIELD OF THE INVENTION

This invention relates to the field of data transmission and more particularly to a method and system for inserting Interval Markers in a block based data transmission system.

BACKGROUND OF THE INVENTION

As the internet and computer networking continue to evolve, data transmission speeds are increasing as well as the amount of data transmitted. The increase in data traffic is occurring in Local Area Networks (LANs) based on Ethernet and other transport mechanisms such as Wide Area Networks (WANs) and Storage Area Networks (SANs) which could use Ethernet or any of a number of data transport mechanisms. Similarly, the amount of data moving through Internet Protocol (IP) based networks such as the internet continues to grow substantially.

Accordingly, users face a growing need for new ways to store and maintain their data. Today's technology offers three basic storage options: Direct Attached Storage (DAS), Network Attached Storage (NAS) and Storage Area Networks (SAN).

In its most basic form, Direct Attached Storage consists of a disk drive directly attached to a personal computer or server. One of the most common methods of transferring data between a hard drive and its associated personal computer or server is the Small Computer Systems Interface (SCSI). Other methods, such as SATA and IDE are well known.

The SCSI protocol uses commands to transfer data as blocks, which are low level, granular units used by storage devices, as opposed to LANs, which typically use file based methods for transferring data. The overall operation and an architectural

description of the SCSI protocol is available from the American National Standards Institute (ANSI), the specific specification having the designation ANSI/INCITS 366-2003, titled *Information Technology – SCSI Architecture Model – 2 (SAM-2)*, herein incorporated reference, and herein referred to as the SCSI Specification.

As internet traffic and storage needs have grown, there is a growing convergence between storage devices, protocols, and IP based transport mechanisms. For example, current SCSI storage devices are designed to work over a parallel cable having a maximum cable length of 12 meters, While IP based transport mechanisms have no data transmission distance limitation.

At the present time, the storage industry and the various industry entities responsible for developing and maintaining the various Internet Protocols are working together to develop standards to enable SCSI based data transfers over the internet. Specifically, the IP Storage (IPS) Working Group of the Internet Engineering Task Force (IEF) is in the process of finalizing a specification for encapsulating SCSI commands in the known TCP/IP protocol. The Internet SCSI (iSCSI) protocol for block storage is predicated on standard Ethernet transports. The iSCSI protocol defines the rules and processes to transmit and receive block storage data over TCP/IP networks.

iSCSI replaces the parallel SCSI direct cabling scheme with a network fabric. iSCSI is transport independent and will support any media that supports TCP/IP. Servers and storage devices that support iSCSI connect directly to an existing IP switch and router infrastructure. iSCSI enables SCSI-3 commands to be encapsulated in TCP packets and delivered reliability over IP networks. The iSCSI specification is complete and undergoing final ratification within the IETF. The current iSCSI specification is available from the IETF under the designation *draft-ietf-ips-iscsi-20.txt*, dated January 19, 2003, and herein referred to as the iSCSI Specification. iSCSI network interfaces under development will be capable of transferring data over the internet in speeds approaching 20 Gbits/sec. The iSCSI protocol is just one example of a network storage protocol, which may employ the Interval Marker System and Method on the present invention, although those skilled in the art will appreciate that the method and system of the present invention is useful in any type of data transfer protocol where Interval Markers are useful or required.

SUMMARY OF THE INVENTION

A System and Method for inserting Interval Markers in a data stream is provided. In one embodiment of the present invention, Interval Markers are inserted between data blocks comprising a data stream transmitted from a storage device to a storage application. A connection between a storage device and the storage application is established wherein the connection is defined by a plurality of parameters, including the number of data blocks to be transmitted and the desired intervals between Interval Markers in the data stream. Data blocks from the storage device are read into a Buffer having a predetermined number of registers. The data blocks are read into the registers in groups of data blocks.

The predetermined number of registers is determined by the number of data blocks within the groups of data blocks and the size of the Buffer includes sufficient registers for simultaneously storing at least first and second groups of data blocks as well as registers for storing Interval Markers.

A Block Counter is initialized at the beginning of the connection for counting the data blocks and is incremented as they are read into the registers. The Block Counter is continuously updated to indicate how many registers in the Buffer contain valid data blocks. A Marker Offset Counter is also initialized at the beginning of the connection, and the Marker Offset Counter is continuously updated to indicate the next location for insertion of an Interval Marker between the data blocks within the data stream. Interval Markers are inserted between data blocks stored in the registers as indicated by the values of the Block Counter and the Marker Offset Counter. The data blocks and Interval Markers are transmitted to the storage application to generate a data stream, when the Block Counter and Marker Offset Counter indicate there is sufficient data in the registers for transmission.

In one embodiment of the present invention, Interval Markers may be used as a Fixed Interval Marker (FIM) as defined in the iSCSI specification, although the present invention may be used in any data transmission scheme where Interval Markers or delimiters are required. The iSCSI specification requires that data blocks are dword

aligned and that Fixed Interval Markers are required at fixed intervals for data flow management. The iSCSI specification does not describe any specific implementation for the creation or insertion of Fixed Interval Markers. It only requires that Fixed Interval Markers may be inserted at predetermined locations relative to the data blocks in the data stream to be transmitted. One method for complying with the iSCSI specification would be to cache all the requested data blocks in memory and to calculate and insert the FIM based on the entirety of data blocks cached in memory. The iSCSI specification requires 32-bit dwords and any given data transmission may include thousands of dwords. In this case a massive amount of memory would be required to cache the entire data transmission. In addition, a significant amount of latency would accumulate while the data is read into memory and the numerous Fixed Interval Markers are calculated and inserted, prior to transmission.

One advantage of the present invention is that it only requires a Buffer having a predetermined size. For example, in the embodiment of the present invention described below, a Buffer having ten (10) 32-bit registers, which store 10 32-bit dwords is shown. A dword is defined as a group of bits constituting a single data block. Depending on the application, dwords can vary in width. For example, dwords can be defined as 8, 16, 32, or 64-bit structures, or even wider, provided they are used consistently within a specific application. The Buffer includes a portion for receiving data, a portion for outputting data, and additional registers for inserting Markers and optimizing data transfers, particularly when Marker insertion occurs during a transmission boundary.

When a data transmission is required, data is read into the Buffer from a Data Storage Module. The data blocks are then managed as they move through the Buffer and are output to a Network Stack when the output portion of the Buffer is filled with valid data blocks. Thus, the present invention can transmit massive amounts of data, without the need for a large data cache. The present invention eliminates latency since data is read into, and read out of, the Buffer on a real time basis. Accordingly, the present invention is particularly useful in streaming data applications.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a diagram showing the protocol stack in a typical Data Network System.

Figure 2 is a diagram of a data structure of typical TCP/IP Data Communication Packet, which may include Interval Markers.

Figure 3 is a system diagram a Data Transmission System which employs the Marker insertion system and method of the present invention.

Figure 4 is a detailed diagram of an exemplary Buffer structure suitable for use in the Data Transmission System of Figure 3.

Figure 5 is a state diagram detailing the operation and use of the Buffer and registers of Figure 4-6.

Figure 6 shows a register model of the Buffer structure of Figure 4 demonstrating various states of the contents of the Buffer structure of Figure 4, during a typical data transmission.

DETAILED DESCRIPTION OF THE INVENTION

The method and system for Marker insertion of the present invention is useful in data transmission systems such as those based on the TCP/IP protocol. Figure 1 shows a Data Network 100 which may employ standard networking protocols such as TCP/IP as well as storage protocols such as SCSI. The Data Network 100 comprises and Initiator System 102 and a Target System 104. The Initiator System 102 includes a Physical Data Link 106 which provides a physical connection to the Internet 108 via any type of physical connection, such as an Ethernet connection common in most Local Area Networks. The Physical Data Link 106 is coupled to a Network Stack 110 which exchanges data with the Physical Data Link 106 in accordance with a Network Communication Protocol such as TCP/IP. The Network Stack 110 is further coupled to a Storage Protocol Services Processor 112 that exchanges data with Network Stack 110. The Storage Protocol Processor 112 processes requests from a Storage Application 114 and encapsulates or decodes packets as requested by Storage Application 114 in accordance with a predetermined data storage protocol such as SCSI.

The Target System 104 includes a set of components that complement those of the Initiator System 102. Specifically, the Target System 104 comprises a Physical Data Link 116, a Network Stack 118, a Storage Protocol Services Processor 120 and a Storage Device Server 122, wherein each of the respective devices in Data Network 100 at each

layer are in logical communication with each other. For example, each of the respective Network Stacks 110, 118 are in cooperative communication through the Physical Data Links 106, 116 to establish and maintain connections, via a Network Communication Protocol such as TCP/IP over the Internet 108, by addressing the appropriate target and destination IP addresses, and opening ports and sockets during an active connection. Similarly, the respective Storage Protocol Services Processors 112, 120 are in logical communication with each other in establishing connections, negotiating parameters and exchanging Data Communication Packets such as those specified in the iSCSI specification. Finally, the Storage Application 114 is in logical communication with the target Storage Device Server 122 in the exchange of data blocks, such as those defined in the SCSI specification.

In operation, the respective Initiator and Target systems 102, 104 operate as typical host and storage devices that are logically coupled with a network connection and through the various service and transport layers below. Thus, any distance limitations imposed by the physical characteristics of the directly connected storage interfaces are eliminated. Further, in many network configurations, Personal Computers, Servers and various Network Attached storage devices will include complementary Target and Initiator Systems. However, the present invention is particularly useful in the context of one device initiating a data communication session with another.

Figure 2 shows an exemplary Data Communication Packet 200 for transmission via TCP/IP according to the iSCSI Specification. As shown, the Data Communication Packet 200 includes an IP Header 202 and a TCP header 206 which are defined in accordance with the industry standard TCP/IP protocol. IP and TCP headers are used in establishing connections and include parameters such as a source address, destination address, and port identification. The TCP/IP protocol also provides for the insertion of an IP checksum 204 between the IP Header 202 and TCP Header 206 that may be used for error correction. Following the TCP Header 204 are a Storage Protocol Header 208 Storage Device Commands 212, and Data Blocks 214, 216. An optional CRC value may be appended to the end of Data Communication Packet 200 for error correction. The Storage Protocol Header 208 may include a number of parameters such as the length of desired Interval Markers, the desired interval between Interval Markers, etc. The storage

device commands include standard commands such as those used in directly attached SCSI systems.

As will be described in greater detail below, Interval Markers 218 – 224 may be inserted in accordance with a predetermined network protocol, such as the one described in the iSCSI Specification, although those skilled in the art will appreciated that Markers may be useful in many applications, where the tracking of specific data blocks is desired.

Since the Network Storage Protocol Header 208, Storage Device Commands 212, and Data Blocks 214, 216 are exchanged between Initiator and Target Systems as blocks within a TCP/IP connection, the physical transport layer becomes somewhat irrelevant. The Network Storage Protocol and Storage Device information appear as nothing more than a string of binary values sent over a physical layer. As such, the entire internet infrastructure is available as a physical transport mechanism for data block transfers.

Referring now to Figure 3, a system diagram of Data Transmission System 300 is shown. Those skilled in the art will appreciate that the Data Transmission System 300 may be implemented in any of a number of ways including implementation entirely in software or hardware, or any combination thereof. As data transmission rates continue to increase, it is becoming increasingly difficult for typical Central Processing Units found in Personal Computers and Servers to manage data traffic without having a negative impact on total system performance. Thus, it is becoming increasingly common for data transmission systems such as those based in the iSCSI Specification to be implemented in devices known as Transmission Offload Engines.

An overview of various Transmission Offload schemes is available from the Storage Networking Industry Association (SNIA). For example, a Whitepaper published by the SNIA IP Storage Forum and entitled *iSCSI Building Blocks for IP Storage Networking* discusses various iSCSI implementations and Transmission Offload Engines. The Data Transmission System 300 is suitable for use as an implementation of Target System 104.

In the Data Transmission System 300, a Data Storage module 302 is used to store data in a host system such as a Personal Computer, Server or Network Storage Device and may include one or several hard disk drives or any type of random access memory. The Data Storage Module 302 is coupled to Data Controller 304 with Memory Control

Bus 306. Data Storage Module 302 is further coupled to Marker Insertion Module 308 through Data Bus 310 and Control Bus 312. Control Bus 312 is used to synchronize transfers of data between the Data Storage Module 302 and Marker Insertion Module 308. Control Block 314 is cooperatively coupled to Marker Insertion Module 308 with Control Bus 316. Control Block 314 is further coupled to Data Controller 304 with Control Bus 318. The specific operation of the various control busses 306, 312, 316 and 318, Data Controller 304, Control Block 314 and Marker Insertion Module 308 is discussed in further detail below.

The output of Marker Insertion Module 308 is coupled to the Network Stack 320 with Data Bus 322 for integration with the Data Communication Packet 200 described in conjunction with Figure 2. Once the Data Communication Packet 200 has been aggregated in Network Stack 320, it is then sent to the Physical Data Link 116 via Data Bus 324.

Referring now to Figure 4, a detailed diagram of Marker Insertion Module 308, and Data Busses 310 and 322 are shown. The interaction of the various control busses and system parameters used during the operation of the present invention are also described. Marker Insertion Module 308 includes a Buffer 402 having a predetermined number of registers, where each register can store a single dword. In the example shown in Figure 4, Buffer 402 utilizes ten (10) registers 404- 422, each of which can store a 32-bit dword, although Buffer 402 could easily be modified to accommodate dwords of any width, or could be modified to have greater depth, for example, in the form of a register queue. A number of parameters affect the overall performance of Marker Insertion Module 308. The width of Data Bus 310 is represented by the parameter (DBin). In the example shown in Figure 4, DBin = 128 bits, or four (4) 32-bit dwords. Thus, four (4) 32-bit dwords can be read into the registers of Buffer 402 in a single clock cycle. The width of Data Bus 310 is represented by the parameter (DBout). In the example shown in Figure 4, DBout = 128 bits, or (4) 32-bit dwords. Thus, four 32-bit dwords can be read out of Buffer 402 in a single clock cycle.

In the example shown, registers 404 - 410 are dedicated for use as Buffer output registers, although if they do not contain valid data, they may be used to input data from Data Bus 310 as well. Additional registers are included to input data from Data Bus 310

and to provide ample room for Marker insertion and register re-ordering, which is discussed in further detail below.

Other parameters used in the operation of Marker Insertion Module 402 include the parameter (Lvi) which indicates the length of valid input data. Lvi has a range between 1 and DBin. In other words, in the present invention, the number of dwords which can be read into Buffer 402 is variable, depending on the width of a dword and the value of DBin. In prior systems, only uniform values are used. The parameter Marker Length (ML) indicates the size of the Marker to be inserted into the data stream. In some cases ML may consist of two adjacent dwords in the event that a Marker spans a data transmission boundary. The variable (MI) indicates the Marker Interval or the distance between Markers. Typically, MI is constant at a predetermined value, although this value may vary for any given connection.

The depth of Buffer 402 is indicated by the parameter (Q) which represents the number of dwords that can fill Buffer 402. While the principles of the present invention can be applied to Buffers of any size, the optimum Q value = DBin + DBout + ML which accounts for data streaming in the worst case scenario while eliminating system deadlocks.

Variables and parameters are managed in the Control Block 314. Data Controller 304 operates in cooperation with Control Block 314 to effect data block transfers as requested by Control Block 314. The value of variable Buffer Count (BC) represents the current number of registers in Buffer 402 containing valid data. The value of BC can range from 0 to Q. In operation, it is initialized at zero the start of a data transfer from host memory, incremented as Buffer 402 is filled, and decremented to zero at the end of each data transfer.

The variable MO or Marker Offset represents how many dwords remain prior to insertion of the next Interval Marker. At the beginning of a data transfer, MO is initialized with the value of MO from the previous transfer. At the end of the data transfer, the last value of MO is stored for use during the next data transfer. The following relationships define the operation of Buffer 402 as data is read into and out of Buffer 402:

At the start of a transfer of data from host memory: BC = 0; and MO = value of MO from the last transfer.

If new input data is read into the Buffer 402:

$$BC\ (new) = (BC(old) + DBin)$$

If data is read out of Buffer 402:

$$BC\ (new) = (BC\ (old) - DBout);\ and$$

$$MO\ (new) = (MO\ (old) - Dbout)$$

If a Marker is inserted into the data stream:

$$BC\ (new) = (BC(old) + ML)\ and$$

$$MO\ (new) = (MO\ (old) + MI)$$

In operation, if new data is present and available in host memory, it is transferred to Buffer 402 over Data Bus 310 on a continuous basis. The variable MO is used as a pointer to indicate which of the registers 404 – 422 constitute the first available register for accepting new data as the registers are filled from left to right. In the example shown, a data transfer will not occur if the variable BC greater than DBin.

Figure 5 shows a state diagram 500 which illustrates the overall operation of Data Transmission System 300. In a quiescent state, Buffer 402 is empty in idle state 502 until Data Controller 304 asserts a signal on Control Bus 3306 that indicates that Data Storage Module 302 should initiate a data transfer to Marker Insertion Module 308. Once a data transfer has been initiated, Data Transmission System 300 enters state 506 which accounts for data block transfers with a variable designated Count_Data. While in state 506, two events are possible. Specifically, the first event occurs if (BC + DBin) is less than or equal to Q, which indicates Buffer 402 has sufficient vacant registers to receive new data. The second event occurs if BC is greater than or equal to DBout, which indicates Buffer 402 has enough valid data to transfer to Network Stack 320. If the variable MO is less than the parameter Q, an Interval Marker insertion is pending and will be inserted somewhere between the data blocks temporarily stored in Buffer 402. Otherwise, Data Transmission System 300 enters state 512, which monitors data traffic with the variable Accum_Data.

In the event (BC + ML) is less than or equal to Q, there are enough vacant registers in Buffer 402 to accommodate Interval Marker insertion and Data Transmission System 300 enters State 512, designated Insert_FIM. In State 514, If Buffer 402 does not have sufficient vacant registers to accommodate Interval Marker insertion, Data Transmission System 300 transitions to State 516 designated Drain_Data. These relationships are summarized as follows:

Transition => State 512: IF $(BC + M) > Q$

Transition => State 516: IF $(BC + ML) \leq Q)$

Transition => State 514: IF $(MO < Q)$ and $(B \geq MO)$ and $((B + MO) \leq Q)$

Transition => State 502: IF DBin = 0

While in State 512, if $((MC + ML) \geq Q)$, enough data has accumulated in registers 404-422 to insert an Interval Marker. If $((BC + M) \leq Q)$, there is sufficient room in Buffer 402 to insert Interval Markers. In this case, a transition to State 514 occurs. Otherwise a transition to State 516 occurs. These relationships are summarized as follows:

Transition => State 514: IF $(BC + ML) \leq Q$

Transition => State 516: IF $(BC+ ML) \leq Q)$

When in State 516, Data Transmission System 300 transfers data in Output registers 404-408 to Network Stack 320 to clear enough register space in Buffer 402 to accommodate the insertion of Interval Markers.

State 516 is characterized as follows:

Transition => State 506: IF $(BC + ML) \leq Q$

State 514 is characterized as follows:

Insert Marker; and

Transition => State 502

Figure 6 shows a typical sequence of data processed by Marker Insertion Module 308 as it passes through Buffer 402. At clock cycle 1, registers 404, 406, 408 and 410 contain valid data, and BC = 4, and MO = 3. Since BC is greater than MO, an Interval Marker is inserted in registers 410, 412 and the prior contents of register 410 are moved to register 414 at clock cycle 2. MO is incremented to 9, reflecting the fact that an Interval Marker has been inserted, and is set to point to the next instance of an Interval Marker. In clock cycle 3, new data is read into registers 416 - 422, respectively and BC is incremented to 10, indicating Buffer 402 is full. In clock cycle 4, the contents of registers 404 - 410 are transferred to Network Stack 320 and the remaining contents of Buffer 402 are right-shifted, thus clearing registers 416 - 422 to accept new data. At the same time, the variable BC is updated to indicate four registers are available and the variable MO is updated to indicate an Interval Marker should be inserted five dwords later. In clock cycle 5, Interval Markers are inserted in registers 414 and 416, respectively, as indicated by the value of MO and the contents of register 418 in clock cycle 4 are shifted to register 418 to accommodate the inserted Interval Markers. Variable BC is incremented to a value of 8 indicating that registers 420, 422 are vacant, and variable MO is updated to a value of 11.

After clock cycle 5, Buffer 402 cannot accept another data transfer, so in clock cycle 6, the contents of registers 404 - 410 are transferred to Network Stack 320 and the contents of Buffer 402 are right-shifted, thus clearing registers 412 - 422. Variable BC is updated to a value of 4 indicating there are 6 available registers in Buffer 402 and Variable MO is updated to a value of 7. In clock cycle 7, four new data packets are read into registers 412 - 418 and variables BC and MO are updated to values of 8 and 7, respectively. In clock cycle 8, Interval Markers are inserted in registers 416-418, respectively and the contents of register 418 in clock cycle 7 are shifted to register 422, to accommodate the inserted Interval Markers, and BC and MO are updated accordingly. The overall pattern continuously cycles until the last data block in a given transmission is reached, as shown at clock cycle 12, wherein register 404 contains a single data block. Once the data block in register 404 is transferred out of Buffer 402, the variables BC and MO are reset to zero (0), indicating a return to idle state 502.

While the various embodiments described above have been described with reference to the iSCSI specification, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with following claims and their equivalents.